

Service Communities: A Structuring Mechanism for Service-Oriented Business Ecosystems[†]

Nirmit Desai², Pietro Mazzoleni³, and Stefan Tai¹

¹IBM Research, USA, e-mail: stai@us.ibm.com

²North Carolina State University, USA, e-mail: nvdesai@ncsu.edu

³University of Milan, Italy, e-mail: mazzoleni@dico.unimi.it

Abstract—As service-oriented computing technologies mature and gain acceptance, the focus of the research community shifts toward applications, business concerns, and business value creation. Service-oriented business ecosystems are one such emerging research area. In our previous work, we argued for a structuring mechanism enabling *service communities* in business ecosystems – dynamic, customizable groups of services provided and used by membership-based social or business networks of varying scale and lifetime. This paper develops the idea of service communities further in three ways. First, we study a variety of motivating business scenarios and articulate the business value of applying the community concept to services. Second, we provide technical details of our middleware platform that supports the runtime creation and management of service communities. Third, we discuss extensions of the general service community idea for further value creation and research.

Index Terms—business ecosystems, service communities, service groups, service marketplaces

I. INTRODUCTION

Business ecosystems and their organizations are facing two major critical challenges related to the use of the Internet as a distribution and integration platform: the introduction of service-oriented architectures for computing, and the use of digital community mechanisms for people collaboration. Services computing technology for the description, discovery, and use of software as services [13] has become a standard programming model and platform. Digital communities, on the other hand, are changing the way people use the Internet for networking purposes and content sharing. We pose the question of how services computing and digital communities together can facilitate business ecosystems design and evolution.

In our recent work [10], we motivated and introduced the concept of *service communities* as a structuring mechanism and middleware for service-oriented business ecosystems. This idea draws from a number of trends and disciplines:

- From the formal paradigm of service-oriented computing, it adopts a standards-based approach for a distributed computing platform and runtime.
- From the informal, internet-based, large-scale social communities (such as facebook.com, flickr.com, and YouTube.com), it adopts membership-based community formation and social community interaction.
- From service marketplaces (such as salesforce.com, jamcracker.com, and reardencommerce.com), it

adopts the notion of selection of services for business solutions from an existing set of available services.

- From ad hoc, per-project, customizable supply chain management models and practices (as exercised by Li and Fung [7]), it adopts the notion of having *small-scale, dynamic* communities of select members for business collaboration.

With this inheritance of properties, we defined the general concept of service communities as a design and runtime abstraction to manage groups of services provided and used by membership-based social or business networks that can be of varying scale and lifetime.

This paper reports on our research progress, developing the idea of service communities (and supporting middleware platform) further and aiming at answering the key question of what it can do for businesses. To that end, Section II describes business scenarios from four diverse categories: service clustering, member clustering, collaboration and project management, contract monitoring and compliance checking.

Section III provides a refined definition of service communities. A taxonomy of such communities is developed and it is shown how each of the above categories can be classified in it. Section IV discusses the details of our middleware prototype which includes an Eclipse UI plug-in for interactive creation and management of service communities. Section V discusses how, with the help of *our prototype*, the basic concept of service communities can readily handle or can easily be extended to handle each of the scenarios of Section II. Extensions of the general concept and middleware are discussed in Section VI.

II. MOTIVATING SCENARIOS

We start with four different business scenarios that motivate the need for a structuring mechanism for service-oriented business ecosystems.

A. Service Clustering

Clusters of machines have been traditionally employed to achieve performance for resource intensive computations and to improve reliability and availability of computational processes via distribution, replication, and load balancing mechanisms. Similarly, there is the need for deployed business services to guarantee a degree of performance and dependability to its customers. In this context, services clustering and grouping mechanisms for common services providers are an approach to improve and ensure client experience.

[†]This research was conducted while Nirmit Desai was an intern and Pietro Mazzoleni was an academic visitor at IBM research.

B. Member Clustering

Businesses often employ a pool of human resources having special skills to resolve service requests from customers. These human resources may form networks for knowledge sharing; tools and techniques enabling and improving expertise sharing are desirable. In addition, efficient utilization of such human resources is a key to their success.

C. Contract Monitoring and Compliance

With business environments tending to be more open than ever, contractual relationships among business partners need to evolve dynamically under a legal context. To address the problem, frameworks for monitoring and verifying the compliance of partners' actions to such contracts have been developed [12]. However, such frameworks depend on the existence of third-party observers who monitor all message exchanges and contracts and blow whistles when contracts are violated. In open business environments, facilitating such an observer is a challenge. One, all partners may not agree on a trusted third party. Two, contracts exist under various legal contexts that affect their management – a single static third-party cannot suit all contracts. Three, all messages have to be routed through the observer which may be technically challenging.

D. Collaboration and Project Management

Supporting collaboration among a diverse and dynamic group of business partners to achieve common business objectives is a known challenge. Examples include

- Designing and managing customizable, per-project supply chains
- Conducting clinical trials in the pharmaceutical industry
- Scheduling and managing field workers to support multiple clients
- Managing product development teams involving parties from operations, sales, marketing, and development departments.

All of these feature the following common properties:

- A diverse group of parties is involved, where parties may join and leave the group dynamically
- The objectives and issues at hand differ from instance-to-instance
- The interactions can be informal and dynamic, if not chaotic.

III. SERVICE COMMUNITIES

Motivated by the above business scenarios and challenges, we propose *Service Communities* as a structuring mechanism to create and manage service-oriented business ecosystems. We define a *Service Community* as a dynamic and evolving group of services and members, where

- Only members can contribute services to the community for use by other members
- Any kind of service can be contributed
- Members play one or more of the roles defined for a community
- Membership criteria can be defined on a per-community per-role basis

Thus, service communities are a controlled services design and runtime mechanism in support of dynamic networks comprising select service providers and users as members. Also, the members may come from within or across, small or large enterprises. Communities allow members to contribute and consume automated (Web) services, and support the collaboration of members through collaboration mechanisms (email, messaging, and Web 2.0 community forums).

Typically, a set of roles and membership criteria are specified while creating a community. The creator automatically plays the role of an administrator. Membership criteria can include the requirement to provide service implementations for one or more community-specific service interfaces.

The main purpose of service communities is to establish a dynamic platform where services of interest to a community are contributed, grouped, consumed, composed, and managed. Each community is a group of services that are contributed by members; additional data management functions and choreography functions for services composition (such as sequencing of invocations) can be provided through the middleware platform.

Service communities also integrate select community practices of Web 2.0, including community-based tagging to provide meta-information for services. Services contributed by members can be tagged using a set of community-specific terms, and matchmaking functions can be applied using these tags. Community management is provided by a middleware platform that supports community creation at any time, termination of existing communities, and community membership management.

This basic runtime platform can support communities of varying composition, scale, and lifetimes. Figure 1 shows a taxonomy (generalization hierarchy) of service communities. A homogeneous services community groups services supporting a common interface whereas a homogeneous member community groups members playing a common role. A heterogeneous community groups diverse services and members. A typical community would feature varying degrees of member and services heterogeneity. However, while support for each of these kinds of communities requires emphasis on various extensions, we believe that our concepts and platform support the most general service communities and can be easily extended to handle these and other specializations.

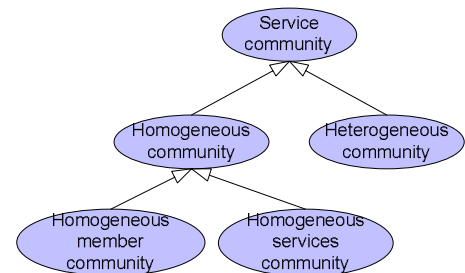


Figure 1: A taxonomy of service communities

Service communities are designed as an extension to standard SOA principles, starting from the basic SOA-triangle of service publication, service discovery, and service invocation involving a service provider, a registry, and a client.

Common infrastructures for business ecosystems (such as service-oriented electronic marketplaces) refine this basic SOA to include a membership model for service providers and clients, and value-add services to mediate, monitor, intercept the services being exchanged. Such infrastructures are further refined by service communities. Whenever an ecosystem infrastructure grows in scale (in terms of number of services and members), additional structuring and runtime grouping mechanisms are needed. Using service communities, membership and other value-add services need no longer be common and applied to each member and each service use, but a membership model and a services directory can now be introduced on a select group basis. Service communities further integrate community functions supporting social community practices such as tagging, rating, and discussion. Community and collaboration functions complement the more formal, standards-based (Web) services computing model with less formal community practices within the same services platform.

In summary, service communities are a design and runtime structuring mechanism that support

- Horizontal, vertical (domain-specific), per-project grouping of services
- Private, trusted services offerings, using a membership-based access control and trust model
- Social networks and interaction practices in services composition, leveraging a spectrum of formal and less formal services description and matchmaking techniques

IV. MIDDLEWARE

We now outline the architecture and design of our middleware prototype (services platform) supporting the concept of service communities.

A. Architecture and Design

As shown in Figure 2, the middleware consists of two sets of middleware services *Core services* and *Per-community services*.

The *core services* support the management of various communities, including creation, termination, and social tagging. These operations are fundamental and independent of the objectives, domain, and members of any particular community.

The *per-community services* support membership management, service management, service provisioning, and information dissemination for a specific community. The middleware supports controlled membership and controlled social tagging to strike a balance between open and informal social community practices and more closed, formal computing networks. At the very basic level, a subject wishing to become a member must meet membership criteria associated to the corresponding community role, such as implementing a specific service interface. Controlled tagging describes a model in which the set of tags that can be applied to a community and contributed services is controlled by the administrators of the community.

Specifically, the core platform supports the following service endpoints:

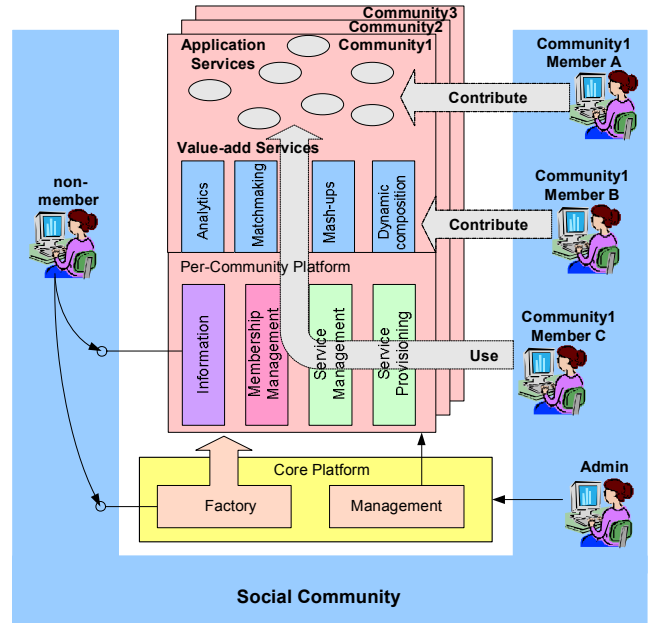


Figure 2: Service communities and the middleware. Non-members can create new, or become members of existing communities. The middleware provides community-independent as well as community-specific management functions

- *Factory*: This is the only public endpoint of the core platform allowing non-members to create new communities by providing a name, a set of roles and corresponding membership criteria, and a set of initial tags. As a result, a new community-specific platform is spawned. Figure 3 illustrates creation of a community. Notice how the creator of the community has access to the Info and Admin endpoints of the newly created community.

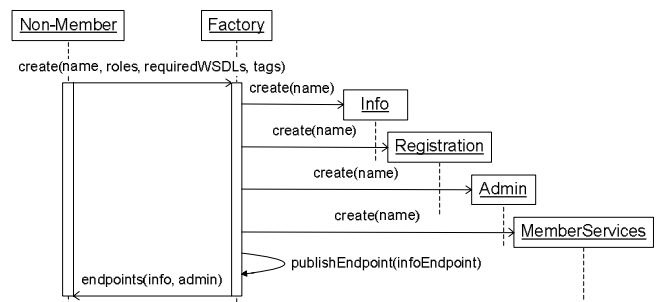


Figure 3: Creation of a community

The per-community platform has the following endpoints:

- *Admin*: Available only to the administrators of a community, this non-public endpoint can be used to terminate the community, or to supply additional applicable tags for the community.
- *Info*: This is the only public endpoint of the per-community platform, providing general information about the community to non-members. Such information may include a summary of the services available, roles and corresponding membership criteria, and the social tags that describe the community. Non-members can apply for membership via this endpoint and may be approved or denied based on the membership criteria. Approved applicants are returned the registration endpoint.

Figure 4 shows a scenario where a non-member joins a

community through a two-step process: inquiry and application through the Info endpoint, and joining through the Registration endpoint.

- **Registration:** Approved applicants can join the community using this endpoint. Separation of approval and joining allows agents to apply, get approved, and share the endpoints with their principals allowing them to join. This endpoint is also used by members who want to leave the community.
- **Member Services:** Members can contribute, consume, tag, and compose services via this endpoint. In addition to application services, value-add services such as analytics and interface and tag-based services matchmaking can also be contributed by members (including administrators) and later be consumed by other members.

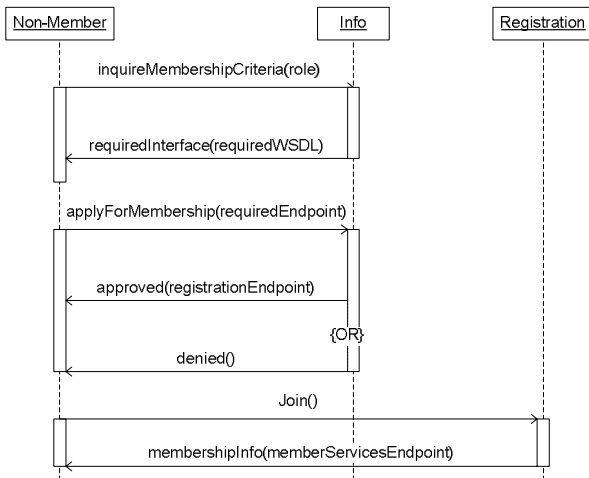


Figure 4: A non-member joining a community

Only the endpoints and meta-information about the contributed application and value-add services are registered with the community; the actual services are hosted elsewhere. This proxy-based approach allows services to exist and be maintained independently by the service providers, and be (potentially) contributed to multiple communities. For platform services that should not be distributed, we are currently developing a model to co-locate and host services on the platform. This approach builds on our previous work on the Cumulus middleware [14].

Referring again to Figure 2, it is possible to see the case where a member A contributes an application service to Community 1 while member B contributes a value-add service; the most important distinction between these two types of services is that value-add services are applied to or injected into the use of application services. Value-add services provide extra functionality such as monitoring, billing, security, reliability, matchmaking, and analytics. Thus, the usage of a contributed application service (as by member C in Figure 2) may be intercepted by one or more selected value-add services. We are investigating the use of a declarative, policy-based model to direct application of value-add services to application services, again building on previous work [11].

As every community is a controlled member environment, and every member can determine which member owns and provides a service, a basic, initial trust model is in place. Additional trust and access control mechanisms, for

example those suggested by the Web services platform, can be used in addition.

Our research prototype is built on open-source components, including Apache Tomcat, Apache Axis2, and Apache Derby, as illustrated below in Figure 5.

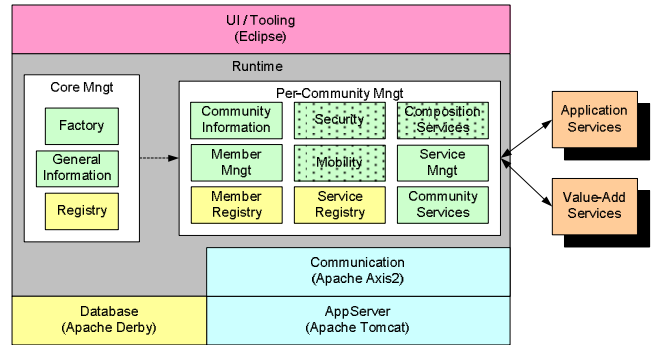


Figure 5: Platform runtime

B. User Interface

To allow interaction with the middleware platform, we have developed an Eclipse UI plug-in. Currently, the plug-in supports interactions such as creating new communities, contributing new services, invoking arbitrary services, sharing credentials for service invocations, and joining and leaving communities. Further, conventional (instant messaging and email) as well as emerging Web 2.0 (tagging and blogging) collaboration support is provided by the platform.

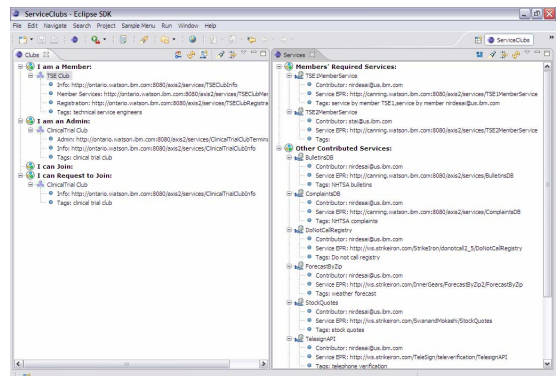


Figure 6: A screenshot of the Eclipse UI

Such an interface can also support Amazon's Mechanical Turk style services that leverage human intelligence for carrying out tasks for which humans are far more effective than machines [1]. Service communities take this concept a step further by scoping the use and integration of Turks to provide customized functionality.

V. APPLICATIONS

This section describes how service communities can be adopted to address the issues described in Section II, and the potential extensions that can be made.

A. Communities for Service Clustering

Service communities can be employed to provide dependability guarantees for business services. Here, the community itself is viewed as a provider of such guarantees instead of just a group of members and service proxies. A few select service endpoints are publicly available for out-

siders to submit requests and receive results. The community can provide a set of middleware services and infrastructure for managing replication, load balancing, and transaction monitoring. In this scenario, members are businesses wishing to provide their services to their clients and partners (typically outside the community) in a reliable manner. Naturally, the administrator can charge the members for the guarantees provided. The platform can support many such infrastructure communities having customized configuration of its resources. According to the taxonomy of Figure 1, such communities possess a higher degree of homogeneity of services. In spirit, this class of communities is related to the service grouping models as suggested in [2], [3].

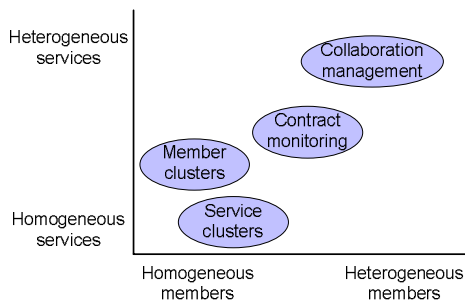


Figure 7: Classification of the scenarios

To support this model, the provisioning of community services would need to be augmented with a billing model for usage of the service replications. Also, the middleware would need a mean to host the service replicas instead of their proxies.

B. Communities for Member Clustering

Similar to service clustering, but in the other dimension, communities can naturally support management of human resources. An enterprise can administer communities for pools of human resources. Such communities would typically involve a role corresponding to the desired skills, which each of the members would play. Membership criteria would include supporting an interface for allowing the managers to issue commands to control the resource pool. Such communities possess a higher degree of homogeneity of members.

An example of such a community is a pool of mobile technical service field engineers managed by automobile manufacturers to resolve technical issues pending at various dealerships that cannot be solved by the dealers themselves.

Our current prototype can support administration of such member groups. If the members are mobile, additional challenges of supporting service interactions over mobile devices from various locations need to be addressed.

C. Communities for Contract Monitoring

Service communities can naturally act as observers and contexts of contracts. A government entity can administer such communities endowed with legal and institutional authority. Partners wishing to engage in contractual relationships can join such a community and interact. The community can provide customizable services for contract management and monitoring. Alternatively, the partners wishing to engage in a contract can contribute their own contract management services and have the administrator as an ob-

server (an interface the administrator supports). As contracts may be enacted under various legal contexts, per-context communities, as against a single legal institution to capture such contexts are a natural solution. The contracts among the partners may include guarantees to exchange services. Such services can be contributed to the community by the involved partners. The criteria for joining such a community would be to support an interface through which the administrator can inform the partners of the status, proactively solicit status from the partners, and certify satisfaction or violation of contracts. According to the taxonomy of Figure 1, such communities manifest a moderate degree of heterogeneity of services due to the heterogeneity of business services involved in contracts, and a moderate degree of heterogeneity of members due to the various partner roles involved in contracts.

Our current prototype can handle such aspects for contract monitoring and compliance. We assume the institutional authorization of the communities can be handled by external means.

D. Communities for Collaboration Management

Although this is the most dynamic and heterogeneous of all the scenarios, such collaborative and project management activities can be naturally supported on a per-project basis. The creator of the community chooses the partners for the project and defines the interfaces expected from each of them. The partners then implement these interfaces and join the community. Other relevant value-add services can be contributed either by the administrators or the members. As the collaboration progresses, members may join or leave. Eventually, at the discretion of the administrators, when the goals of the project are achieved, the community may be terminated. According to the taxonomy of Figure 1, such communities manifest a higher degree of service heterogeneity and a higher degree of member heterogeneity.

An example of such a community is the management of per-drug trials executed by a pharmaceutical company [5]. In a highly dynamic environment, the company, hospitals, physicians, and clinical research organizations collaborate to carry out the trials. The actual collaboration depends on the type of drug, the protocols to be followed, the regulations in place, and the target market.

Our current middleware prototype supports such collaborations, not by means of providing the actual services, but by allowing the collaborators to share such domain specific services and human expertise and knowledge.

VI. SUMMARY AND DIRECTIONS

We described four distinct motivating business scenarios and discussed how our middleware platform and concept of service communities can help support them. We also provided details on the middleware prototype and a user interface. The concept is deliberately kept simple, but designed for extensibility, which we discuss below.

A. Relationships among Communities

It is common for real-life communities and organizations to be related to each other; one maybe part of another or one may depend on another to achieve its objectives, and so on. Our conceptual model can be extended to allow communi-

ties to be members of other communities. These can be classified as hierarchical communities – a new addition to our taxonomy. A side effect of this extension is that organizations can also be modelled in terms of service communities. When communities themselves act as members, they enter into contracts, have goals, and act proactively to achieve them. Notice how this is in contrast to the passive groups of services and members.

B. Community-based Services Composition

Several approaches have been proposed to address services composition. A common missing element in these approaches is the support for involvement of humans and tools for allowing human designers to compose services. For this reason, services communities integrate research on *Swashup*, an ongoing research project at IBM. *Swashup* provides rich, AJAX-based GUI tools and underlying techniques for not only building composite services, but also recording and sharing the steps and human activities carried out to build them from existing services, as *recipes*. A community can provide a meaningful scope for the services available for composition. As the community members share a common folksonomy and common interests, services contributed are potentially better candidates for composition than all kinds of services available on the Web. Another advantage is that the expertise gained by building composite services in one community, in the form of recipes, can be shared with other communities via the platform.

C. Middleware and Value-Add Services

Composition of business services (applications) has received a lot of attention from the community. But, supporting business ecosystems also requires composition of various *value-add services* (such as billing, accounting) and *middleware services* (such as transactions, security) supporting e-commerce and technical infrastructure and interoperability concerns. While business service composition calls attention to data mediation and choreography mediation issues, composition of value-add and middleware services requires special techniques such as layering and injection [14]. From a middleware viewpoint, we plan to support such compositions by leveraging phases and handlers as supported in the Axis2 middleware; these must be complemented with a business model and mechanisms supporting the model, however, if value-add and middleware services are offerings on an open market.

VII. RELATED WORK

Our work is motivated by complex business applications and scenarios as described in Section II and draws positives from the trends in services computing, social computing, customizable supply chain collaborations, and service marketplaces.

The concept of *Service Communities* applies to electronic marketplaces [8] and service-oriented business process management [6], however, aims to introduce increased flexibility and dynamics in the services managed and provided by marketplaces and used in business processes. This objective is in line with research on adaptive middleware, an example of which is the Cumulus architecture that supports an on-demand middleware-as-services model [14].

From a technical viewpoint, service communities are related to group concepts in distributed computing. A variety of group concepts exist; these are mostly concerned with server replicas, addressing load-balancing, security, and fault-tolerance through coordination and consensus protocols [4]. Service communities are not limited to groups of homogeneous services, but naturally extend to support heterogeneity of members and services. Service communities further integrate (less formal) community and human collaboration practices for services computing, exploiting community knowledge, expertise, and skills.

VIII. REFERENCES

- [1] Amazon Mechanical Turk. <http://www.mturk.com/mturk/welcome>
- [2] B. Benatallah, M. Dumas and Q. Sheng. Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. *Distributed and Parallel Databases* 17(1), 2005
- [3] S. Dustdar and M. Treiber. VISCO – Vienna Service Communities. <http://www.vitalab.tuwien.ac.at/projects/visco/>
- [4] R. Guerraoui, P. Felber, B. Garbinato and K. Mazouni. System Support for Object Groups. In *Proceedings of the ACM Conference on Object-oriented Programming Systems, Languages and Applications (OOPSLA'98)*, 1998
- [5] IBM Business Consulting Services. *Pharma 2010: The threshold of innovation*. February 2006. Available online at <http://www1.ibm.com/services/us/index.wss/ibvstudy/imc/a1001099>
- [6] R. Khalaf, A. Keller and F. Leymann. *Business Processes for Web Services: Principles and Applications*. IBM Systems Journal 45(2), 2006
- [7] J. Magretta. Fast, Global, and Entrepreneurial: Supply Chain Management, Hong Kong Style. *Harvard Business Review*, September 1998
- [8] J. Sairamesh, R. Mohan, M. Kumar, L. Hasson, and C. Bender. A platform for business-to-business sell-side, private exchanges and marketplaces. *IBM Systems Journal* 41(2), 2002
- [9] J. Spohrer and D. Riecken. *Services Science*. *Communications of the ACM* 49(7), 2006
- [10] S. Tai, N. Desai, and P. Mazzoleni. *Service Communities: Applications and Middleware*. *International Workshop on Software Engineering and Middleware, 2006* (to appear, ACM DL). Available online at <http://www4.ncsu.edu/~nvdesai/sem06.pdf>
- [11] S. Tai, R. Khalaf, and T. Mikalsen. *Composition of Coordinated Web Services*. In *Proceedings of Middleware 2004*, Springer LNCS 3231, 2004
- [12] M. Venkatraman and M. P. Singh. *Verifying Compliance with Commitment Protocols: Enabling Open Web-Based Multiagent Systems*. *Autonomous Agents and Multiagent Systems* 2(3), 1999
- [13] S. Weerawarana, F. Curbera, F. Leymann, T. Storey and D. Ferguson. *Web Services Platform Architecture*. Prentice-Hall, 2005
- [14] E. Wohlstadt, S. Tai, T. Mikalsen, J. Diament and I. Rouvellou. *A Service-oriented Middleware for Runtime Web Services Interoperability*. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2006)*, 2006